We Claim:

1. A method of type-checking a programming language in a compiler according to one or more rule sets comprising:

selecting one or more of the rule sets based upon the present stage of compilation; and

type-checking the programming language based on the selected one or more rule sets.

2. The method of claim 1 wherein selecting one or more of the rule sets is based upon a characteristic of the programming language rather than the stage of compilation.

3. The method of claim 2 wherein the characteristic of the programming language describes the type system of the language.

4. The method of claim 1 wherein type-checking the programming language comprises type-checking each of a plurality of intermediate representations of the programming language.

5. The method of claim 4 wherein the selected one or more rule sets are different for each representation.

6. The method of claim 4 wherein selecting one or more of the rule sets is based upon a characteristic of the representation being type-checked rather than the stage of compilation.

7. The method of claim 1 wherein the programming language includes a type that indicates an element of the programming language can be one of a plurality of types.

8. The method of claim 7 wherein the one or more rule sets contains rules for type-checking a type that indicates an element of the programming language can be one of a plurality of types.

9. The method of claim 1 wherein the one or more rule sets comprise a plurality of rules in a hierarchical format.

10. The method of claim 1 wherein the one or more rule sets comprise one rule set corresponding to strong type-checking, one corresponding to weak type-checking, and one corresponding to representation type-checking.

11. The method of claim 10 wherein the representation type-checking will allow dropped type information for elements of the programming language.

12. The method of claim 10 wherein the weak type-checking will allow type casting.

13. A compiler for compiling source code written in a source language comprising:

a plurality of type rules;

a type-checker that selects a subset of type rules from the plurality of type rules based upon the source language to construct a rule set.

14. The compiler of claim 13 wherein the subset of type rules is selected based upon an intermediate representation of the source code rather than the source language.

15. The compiler of claim 13 wherein subset of type rules is selected based upon a stage of compilation rather than the source language.

16. The compiler of claim 13 wherein three rule sets are constructed: one corresponding to strong type-checking, one corresponding to weak type-checking, and one corresponding to representation type-checking.

17. The compiler of claim 16 wherein the representation type-checking will allow dropped type information for elements of the source code.

18. The compiler of claim 16 wherein the weak type-checking will allow type casting.

19. A type-checking system for type-checking source code authored in a plurality of source languages comprising:

a plurality of rule sets; and

a type-checker, wherein the type-checker selects one or more rule sets to apply to the source code at each of a plurality of representations.

20. The system of claim 19 wherein the selected one or more rule sets is selected based on one or more characteristics of the source language.

21. The system of claim 19 wherein the type-checker is part of a compiler.

22. The system of claim 19 wherein each of the plurality of rule sets corresponds to a specific source language.

23. The system of claim 19 wherein each of the plurality of rule sets corresponds to a specific strength of type-checking.

24. A method of analyzing source code represented as a typed intermediate representation in a compiler comprising:

selecting a rule set from among different rule sets based on the typed intermediate representation; and

analyzing the typed intermediate representation based on the selected rule set.

25. The method of claim 24 wherein selecting a rule set is based on which step in a series of steps in a compilation process produced the typed intermediate representation.

26. The method of claim 24 wherein selecting a rule set is based on a source language from which the typed intermediate representation was produced.

27. The method of claim 24 wherein selecting a rule set is based on a processor architecture.

28. The method of claim 24 wherein selecting a rule set is based on a whether the typed intermediate representation represents verified or unverified code.

29. The method of claim 24 wherein selecting a rule set is based on a whether the typed intermediate representation represents type-safe code or code that is not type-safe.

30. The method of claim 24 wherein selecting a rule set is based on a whether the typed intermediate representation represents typed or untyped code.

31. The method of claim 24 wherein selecting a rule set is based on a whether the typed intermediate representation represents strongly or weakly typed code.

32. A computer-readable medium containing computer-executable instructions for implementing the method of claim 24.

33. A programming interface comprising:

a means for constructing a plurality of rules for checking the consistency of an intermediate representation of a program, wherein checking the consistency of the intermediate representation of a program comprises providing a plurality of rules to a type-checker operable to apply a first set of the plurality of rules to a first intermediate representation, and a second set of the plurality of rules to a second intermediate representation.

34. A computer-readable medium containing computer-executable instructions for implementing the method of claim 1.